Docket No. AUS920040063US1 *PATENT*

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: **Dimpsey et al.**	§	
	§	Group Art Unit: **2192**
Serial No. **10/806,917**	§	
	§	Examiner: **Wang, Ben C.**
Filed: **March 22, 2004**	§	
	§	
For: **Method and Apparatus for Autonomic Test Case Feedback Using Hardware Assistance for Data Coverage**	§	

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

35525
PATENT TRADEMARK OFFICE
CUSTOMER NUMBER

APPEAL BRIEF (37 C.F.R. 41.37)

This brief is in furtherance of the Notice of Appeal, filed in this case on December 19, 2007.

A fee of $510.00 is required for filing an Appeal Brief. Please charge this fee to IBM Corporation Deposit Account No. 09-0447. No additional fees are believed to be necessary. If, however, any additional fees are required, I authorize the Commissioner to charge these fees which may be required to IBM Corporation Deposit Account No. 09-0447. No extension of time is believed to be necessary. If, however, an extension of time is required, the extension is requested, and I authorize the Commissioner to charge any fees for this extension to IBM Corporation Deposit Account No. 09-0447.

REAL PARTY IN INTEREST

The real party in interest in this appeal is the following party: International Business Machines Corporation of Armonk, New York.

RELATED APPEALS AND INTERFERENCES

This appeal has no related proceedings or interferences.

STATUS OF CLAIMS

A. TOTAL NUMBER OF CLAIMS IN APPLICATION

The claims in the application are: 1-26

B. STATUS OF ALL THE CLAIMS IN APPLICATION

Claims canceled: 2, 4-7, 12, 14-17, 22 and 24-26

Claims withdrawn from consideration but not canceled: None

Claims pending: 1, 3, 8-11, 13, 18-21 and 23

Claims allowed: None

Claims rejected: 1, 3, 8-11, 13, 18-21 and 23

Claims objected to: None

C. CLAIMS ON APPEAL

The claims on appeal are: 1, 3, 8-11, 13, 18-21 and 23

STATUS OF AMENDMENTS

An Amendment after the Final Office Action dated October 4, 2007, was not filed. Accordingly, the claims on appeal herein are as amended in the Response to Office action dated July 20, 2007.

SUMMARY OF CLAIMED SUBJECT MATTER

A. CLAIM 1 - INDEPENDENT

The subject matter of claim 1 is directed to a method in a data processing system for presenting coverage data relating to data access occurring during execution of code. The coverage data containing data access indicators associated with memory locations is obtained (Step **2300**, **Figure 23**; page 53, lines 18-19). The data access indicators that have been set by a processor in the data processing system in response to access of the memory locations during execution of the code by the processor is identified to form set data access indicators (page 53, lines 20-24), wherein each set data access indicator is associated with a portion of the memory locations allocated for the code, and wherein the data access indicators are located in one of a shadow memory (**705**, **Figure 7**, page 31, lines 11-13); or a page table (**830**, **Figure 8**; page 33, lines 9-25). Unset data access indicators that have remained unset during execution of the code by the processor are identified (page 41, lines 20-31), and a presentation for coverage data is generated in response to an event for identifying memory locations that have been accessed and for identifying memory locations that have not been accessed during execution of the code (Step **2302**, **Figure 23**; page 53, lines 25-26; page 41, lines 2-7), wherein the set data access indicators and the unset data access indicators are identified in the presentation by one of using a first color to identify the set data access indicators and using a second color to identify the unset data access indicators (page 42, lines 2-7), and using a graphical indicator to identify the set data access indicators and the unset data access indicators (page 42, lines 7-12).

B. CLAIM 11 - INDEPENDENT

The subject matter of claim 11 is directed to a data processing system for presenting coverage data relating to data access occurring during execution of code. The data processing system includes obtaining means (**1400**, **Figure 14**; page 41, lines 13-15) for obtaining the coverage data containing data access indicators (**1402**, **Figure 14**; page 41, lines 13-15) associated with memory locations, first identifying means (**1400**, **Figure 14**; page 41, lines 20-28) for identifying the data access indicators that have been set by a processor in the data processing system in response to access of the memory locations during execution of the code by the processor to form set data access indicators, wherein each set data access indicator is associated with a portion of the

memory locations allocated for the code, and wherein the data access indicators are located in one of a shadow memory (**705**, **Figure 7**; page 31, limes 11-13) or a page table (**830**, **Figure 8**; page 33, lines 9-25), and second identifying means (**1400**, **Figure 14**; page 41, lines 20-31) for identifying unset data access indicators that have remained unset during execution of the code by the processor (page 41, lines 20-31). The data processing system further includes generating means (**1400**, **Figure 14**; page 42, lines 2-12) for generating a presentation for coverage data (**2200**, **Figure 22**; page 52, lines 25-28) in response to an event for identifying memory locations that have been accessed and for identifying memory locations that have not been accessed during execution of the code, wherein the set data access indicators and the unset data access indicators are identified in the presentation by one of using a first color to identify the set data access indicators and using a second color to identify the unset data access indicators (page 42, lines 2-7), and using a graphical indicator (**1406**, **Figure 14**; page 42, lines 2-3) to identify the set data access indicators and the unset data access indicators.

C. CLAIM 21 - INDEPENDENT

The subject matter of claim 21 is directed to a computer program product in a recordable-type computer readable medium for presenting coverage data relating to data access occurring during execution of code. The computer program product includes first instructions for obtaining the coverage data containing data access indicators associated with memory locations (Step **2300**, **Figure 23**; page 53, lines 18-19), second instructions for identifying the data access indicators that have been set by a processor in the data processing system in response to access of the memory locations during execution of the code by the processor to form set data access indicators (page 53, lines20-24), wherein each set data access indicator is associated with a portion of the memory locations allocated for the code, and wherein the data access indicators are located in one of a shadow memory (**705**, **Figure 7**; page 31, lines 11-13) or a page table (**830**, **Figure 8**; page 33, lines 9-25), and third instructions for identifying unset data access indicators that have remained unset during execution of the code by the processor. The computer program product further includes fourth instructions for generating a presentation for coverage data in response to an event for identifying memory locations that have been accessed and for identifying memory locations that have not been accessed during execution of the code (Step **2302**, **Figure 23**; page 53, lines 25-26,

page 41, lines 2-7), wherein the set data access indicators and the unset data access indicators are identified in the presentation by one of using a first color to identify the set data access indicators and using a second color to identify the unset data access indicators (Page 42, lines 2-7), and using a graphical indicator to identify the set data access indicators and the unset data access indicators (page 42, lines 7-12).

D. CLAIM 8 - DEPENDENT

The subject matter of claim 8, which depends from claim 1 and recites that the event is at least one of a completion of the execution of the code, expiration of a time, and the execution of a selected type of instruction in the code (page 5, lines 1-17).

E. CLAIM 9- DEPENDENT

The subject matter of claim 9, which depends from claim 1, recites that the portion of the memory locations is a single memory location in the code and wherein every memory location in the memory locations is associated with a different data access indicator (page 24, lines 7-12).

F. CLAIM 18-DEPENDENT

The subject matter of claim 18, which depends from claim 11, recites that the event is at least one of a completion of the execution of the code, expiration of a time, and the execution of a selected type of instruction in the code (page 5, lines 1-17).

G. CLAIM 19-DEPENDENT

The subject matter of claim 19, which depends from claim 18, recites that the portion of the memory locations is a single memory location in the code and wherein every memory location in the memory locations is associated with a different data access indicator (page 24, lines 7-12).

GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

The ground of rejection to review on appeal is as follows:

A. GROUND OF REJECTION 1

Claims 1, 3, 8-11, 13, 18-21 and 23 are rejected under 35 U.S.C. § 103(a) as being unpatentable over Gover et al., U.S. Patent No. 5,752,062 (hereinafter "Gover") in view of Johnston et al., U.S. Patent No. 6,212,675 B1 (hereinafter "Johnston").

ARGUMENT

A. GROUND OF REJECTION 1 (Claims 1, 3, 8-11, 13, 18-21 and 23)

Claims 1, 3, 8-11, 13, 18-21 and 23 are rejected under 35 U.S.C. § 103(a) as being unpatentable over Gover in view of Johnston.

A.1. Claims 1, 3, 8-11, 13, 18-21 and 23

In finally rejecting the claims, the Examiner states with respect to claim 1:

> 3. **As to claim 1** (Currently Amended), Gover discloses a method in a data processing system for presenting coverage data relating to data access occurring during execution of code, the method comprising:
>
> - obtaining the coverage data containing data access indicators associated with memory locations (e.g., Fig. 5 - elements "63" - implement selected performance monitoring, "65" - collect selected performance monitoring data; Fig. 6A - Monitor Mode Control Register 0 (*MMCRO*); Col. 1, Lines 52-58 - Understanding the memory hierarchy behavior aids in developing algorithms that schedule and/or partition tasks, as well as distribut4 and structure data for optimizing the system; Lines 60-66 - The performance monitor produces information relating to the utilization of a processor's instruction execution and storage' control);
> - identifying the data access indicators that have been set (e.g., Col. 8, Lines 42- 50 - even .. to be recorded/counted, counter .. selection, counter freeze; Col. 10, Lines 53-63; Col. 11, Lines 14-50) by a processor in the data processing system in response to access of the memory locations during execution of the code by the processor to form set data access indicators, wherein each set data ' instruction access indicator is associated with a portion of the memory locations allocated for the code (e.g., Fig. 3 - FINISHED; Col. 20, Lines 48-65 - number ... branches dispatched .. completed; Col. 22, Lines10 -35 - *load* or store; Fig. 8 - a typical superscalar pipeline), and wherein the data access indicators are located in one of a shadow memory or a page table (e.g., Col. 8, Lines 26-39 - Note: special registers with state or content - *MMCRn* (Monitor Mode Control Registers) - maintained via special privilege access mode and being kept in parallel with execution scheduling - see Col. 11, Lines 14-50 - as informational support thereof, hence reads on shadowing type of information kept in memory; see Col. 9, Lines 36-57 - SSR (Saved State Register));

- identifying unset data access indicators that have remained unset during execution of the code by the processor (e.g., Col. 12, Lines 10-11 - once the enabled *IABR* match occurs, bit 29 of *MMCRI* is reset and counting occurs; Lines 19-21 - when an enabled *IABR* match occurs, hardware rests *PCUIABR* to zero and counting is enabled).

Gover does not explicitly disclose generating a presentation for coverage data in response to an event for identifying memory locations that have been accessed and for identifying memory locations that have not been accessed during execution of the code, wherein the set data access indicators and the unset data access indicators are identified in the presentation by one of using a first color to identify the set data access indicators and using a second color to identify the unset data access indicators, and using a graphical indicator to identify the set data access indicators and the unset data access indicators.

However, in an analogous art of Presentation of Visual Program Test Coverage Information, Johnston discloses generating a presentation for coverage data in response to an event for identifying memory locations that have been accessed and for identifying memory locations that have not been accessed during execution of the code, wherein the set data access indicators and the unset data access indicators are identified in the presentation by one of using a first color to identify the set data access indicators and using a second color to identify the unset data access indicators, and using a graphical indicator to identify the set data access indicators and the unset data access indicators (e.g., Col. 1, Lines 62 through Col. 2, Lines 15 - "Optionally, the means for visually indicating further comprises using a first color for each of the elements having the element test coverage status which indicates covered, and a second color for each of the elements having the element test coverage status which indicates not covered" (emphasis added)).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Gover into the Johnston's system to further provide generating a presentation for coverage data in response to an event for identifying memory locations that have been accessed and for identifying memory locations that have not been accessed during execution of the code, wherein the set data access indicators and the unset data access indicators are identified in the presentation by one of using a first color to identify the set data access indicators and using a second color to identify the unset data access indicators, and using a graphical indicator to identify the set data access indicators and the unset data access indicators in Gover system.

The motivation is that it would further enhance the Gover's system by taking, advancing andlor incorporating Johnston's system which offers significant advantages for providing a technique for presenting test coverage metrics for a visual program in a manner that is familiar to the visual

programmer, and providing this additional information using color, line thickness, line style, added text, or other graphical representations as once suggested by Johnston (e.g., Summary of the Invention, Lines 29-43).

Final Office Action dated October 4, 2007, pages 3-6. (emphasis in original)

Claim 1 on appeal herein is as follows:

1. A method in a data processing system for presenting coverage data relating to data access occurring during execution of code, the method comprising:

obtaining the coverage data containing data access indicators associated with memory locations;

identifying the data access indicators that have been set by a processor in the data processing system in response to access of the memory locations during execution of the code by the processor to form set data access indicators, wherein each set data access indicator is associated with a portion of the memory locations allocated for the code, and wherein the data access indicators are located in one of a shadow memory or a page table;

identifying unset data access indicators that have remained unset during execution of the code by the processor; and

generating a presentation for coverage data in response to an event for identifying memory locations that have been accessed and for identifying memory locations that have not been accessed during execution of the code, wherein the set data access indicators and the unset data access indicators are identified in the presentation by one of using a first color to identify the set data access indicators and using a second color to identify the unset data access indicators, and using a graphical indicator to identify the set data access indicators and the unset data access indicators.

The Examiner bears the burden of establishing a *prima facie* case of obviousness based on prior art when rejecting claims under 35 U.S.C. § 103. *In re Fritch*, 972 F.2d 1260, 23 U.S.P.Q.2d 1780 (Fed. Cir. 1992). The prior art reference (or references when combined) must teach or suggest all the claim limitations. *In re Royka,* 490 F.2d 981, 180 USPQ 580 (CCPA 1974). In determining obviousness, the scope and content of the prior art are… determined; differences between the prior art and the claims at issue are… ascertained; and the level of ordinary skill in the pertinent art resolved. Against this background the obviousness or non-obviousness of the subject matter is determined. *Graham v. John Deere Co.*, 383 U.S. 1 (1966). "Often, it will be necessary for a court to look to interrelated teachings of multiple patents; the effects of demands known to the design community or present in the marketplace; and the background knowledge possessed by a person

having ordinary skill in the art, all in order to determine whether there was an apparent reason to combine the known elements in the fashion claimed by the patent at issue." *KSR Int'l. Co. v. Teleflex, Inc.*, No. 04-1350 (U.S. Apr. 30, 2007). "*Rejections on obviousness grounds cannot be sustained by mere conclusory statements; instead, there must be some articulated reasoning with some rational underpinning to support the legal conclusion of obviousness.* Id. (citing *In re Kahn*, 441 F.3d 977, 988 (CA Fed. 2006))."

Appellants respectfully submit that no *prima facie* obviousness rejection can be stated against claim 1 because neither Gover nor Johnston, considered alone or in combination, teaches or suggests all of the features recited in claim 1.

Initially, neither Gover nor Johnston relates to the subject matter to which claim 1 is directed, namely, a method for "presenting coverage data relating to data access occurring during execution of code." Instead, and as will be described more fully hereinafter, Gover is directed to performance monitoring in processing systems such as monitoring an order of processor events during execution of code, and Johnston is directed to presenting coverage metrics relating to testing software.

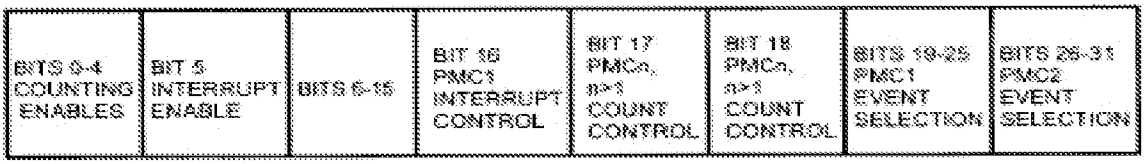
Further, Appellants respectfully submit that neither Gover nor Johnston nor their combination teaches or suggests any of the claim 1 features of "obtaining the coverage data containing data access indicators associated with memory locations", "identifying the data access indicators that have been set by a processor in the data processing system in response to access of the memory locations during execution of the code by the processor to form set data access indicators, wherein each set data access indicator is associated with a portion of the memory locations allocated for the code, and wherein the data access indicators are located in one of a shadow memory or a page table", "identifying unset data access indicators that have remained unset during execution of the code by the processor", or "generating a presentation for coverage data in response to an event for identifying memory locations that have been accessed and for identifying memory locations that have not been accessed during execution of the code, wherein the set data access indicators and the unset data access indicators are identified in the presentation by one of using a first color to identify the set data access indicators and using a second color to identify the unset data access indicators, and using a graphical indicator to identify the set data access indicators and the unset data access indicators."

As indicated above, Gover is directed to performance monitoring in processing systems, and especially to monitoring an order of processor events during execution of code (see column 1, lines 43-46 of Gover). Gover discloses the use of at least one performance monitor counter (PMC) and at least one monitor mode control register (MMCR) to configure the operation of the PMC (see column 2, lines 36-67 of Gover). Gover discloses counting events to identify false triggering, identify bottlenecks, monitor stalls and idles and the like (see column 3, line 53-column 4, line 6 of Gover).

In rejecting claim 1, the Examiner refers to column 1, lines 52-58 and 60-66 and to Figures 5 and 6A of Gover as disclosing "obtaining the coverage data containing data access indicators associated with memory locations." Column 1, lines 50-66 of Gover is reproduced below for the Board's convenience:

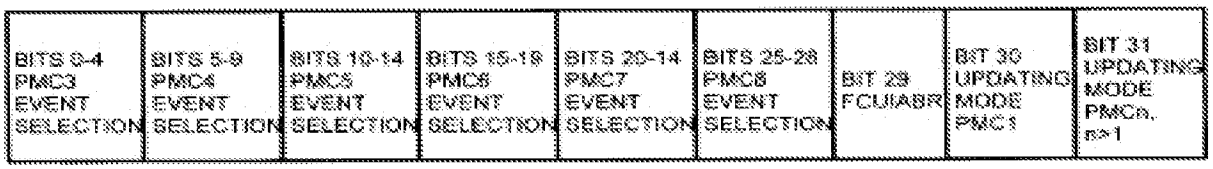
> In typical computer systems utilizing processors, system developers desire optimization of execution software for more effective system design. Usually, studies of a program's access patterns to memory and interaction with a system's memory hierarchy are performed to determine system efficiency. Understanding the memory hierarchy behavior aids in developing algorithms that schedule and/or partition tasks, as well as distribute and structure data for optimizing the system.
>
> Performance monitoring is often used in optimizing the use of software in a system. A performance monitor is generally regarded as a facility incorporated into a processor to monitor selected characteristics to assist in the debugging and analyzing of systems by determining a machine's state at a particular point in time. Often, the performance monitor produces information relating to the utilization of a processor's instruction execution and storage control.

Figure 5 of Gover is a flowchart that illustrates an over all process flow that includes performance monitoring. As described in column 9, line 58-column 10, lime 16, the process includes implementing performance monitoring (Step 63) and collecting performance monitoring data (step 65). Figure 6A is an example of a configuration of a monitor mode control register (MMCRO) for controlling the operation of two PMC counters Figures 6A and 6B are reproduced below for the convenience of the Board:



MONITOR MODE CONTROL REGISTER 0
(MMCR0)

FIGURE 6A



MMCR1

FIGURE 6B

Figure 6A is described in column 10, lines 40-47 of Gover as follows:

> In FIG. **6a**, an example representation of one configuration of MMCRO suitable for controlling the operation of two PMC counters, e.g., PMC1 and PMC2, is illustrated. As shown in the example, MMCRO is partitioned into a number of bit fields whose settings select events to be counted, enable performance monitor interrupts, specify the conditions under which counting is enabled, and set a threshold value (X).

Column 1, lines 52-58 and 60-66 of Gover referred to by the Examiner and reproduced above, do not disclose or suggest "obtaining the coverage data containing data access indicators associated with memory locations" as recited in claim 1. To the contrary, the recitations merely describe the use of performance monitors to count events and the setting of bit fields in a monitor mode control register to select the events that are to be counted. The bit fields recited in Gover are not the same as the "data access indicators" recited in claim 1, and are certainly not "data access indicators associated with memory locations" as recited in claim 1.

Gover also does not disclose or suggest "identifying the data access indicators that have been set by a processor in the data processing system in response to access of the memory locations during execution of the code by the processor to form set data access indicators" as recited in claim 1. The Examiner refers to column 10, lines 42-50 and lines 53-63 and to column 11, lines 14-50 of

Gover as disclosing "data access indicators that have been set by a processor", and to column 20, lines 48-65 and column 22, lines 10-35 as disclosing that the data access indicators are set "in response to access of memory locations during execution of code to form set data access indicators." Appellants respectfully disagree. Column 10, lines 40-63 and column 11, lines 14-50 of Gover are reproduced below:

> In FIG. **6a**, an example representation of one configuration of MMCRO suitable for controlling the operation of two PMC counters, e.g., PMC1 and PMC2, is illustrated. As shown in the example, MMCRO is partitioned into a number of bit fields whose settings select events to be counted, enable performance monitor interrupts, specify the conditions under which counting is enabled, and set a threshold value (X).
>
> The threshold value (X) is both variable and software selectable and its purpose is to allow characterization of certain data, such that by accumulating counts of accesses that exceed decreasing threshold values, designers gain a clearer picture of conflicts. The threshold value (X) is considered exceeded when a decrementer reaches zero before the data instruction completes. Conversely, the threshold value is not considered exceeded if the data instruction completes before the decrementer reaches zero. Of course, depending on the data instruction being executed, completed has different meanings. For example, for a load instruction, "completed" indicates that the data associated with the instruction was received, while for a "store" instruction, "completed" indicates that the data was successfully written. A user readable counter, e.g., PMC1, suitably increments every time the threshold value is exceeded.

Gover, Column 10, lines 40-63

> For example, bit **1** is a freeze counting while in a supervisor state (FCS) bit, bit **2** is a freeze counting while in a problem state (FCP) bit, bit **3** is a freeze counting while PM=1 (FCPM1) bit, and bit **4** is a freeze counting while PM=0 (FCPMO) bit. PM represents the performance monitor marked bit, bit **29**, of a machine state register (MSR) (an SPR **40**, FIG. 1). For bits **1** and **2**, a supervisor or problem state is indicated by the logic level of the PR (privilege) bit of the MSR. The states for freezing counting with these bits are as follows: for bit **1**, FCS=1 and PR=0; for bit **2**, FCP=1 and PR=1; for bit **3**, FCPM1=1 and PM=1; and for bit 4, FCPMO=1 and PM=0. The state for allowing counting with these bits are as follows: for bit **1**, FCS=1 and PR=1; for bit **2**, FCP=1 and PR=0; for bit **3**, FCPM1=1 and PM=0; and for bit **4**, FCPMO=1 and PM=1.
>
> Bits **5**, **16**, and **17** are utilized to control interrupt signals triggered by PMCn. Bits **6-9** are utilized to control the time or event-based transitions. The threshold value (X) is variably set by bits **10-15.** Bit **18** control counting enablement for PMCn, n>1, such that when low, counting is enabled, but when high, counting is disabled until bit 0 of PMC1 is high or a performance monitoring exception is signaled. Bits **19-25** are used for event selection, i.e, selection of signals to be counted, for PMC1.
>
> FIG. **6b** illustrates a configuration of MMCR1 in accordance with a preferred embodiment of the present invention. Bits **0-4** suitably control event selection for PMC3,

while bits **5**-9 control event selection for PMC4. Similarly, bits **10**-**14** control event selection for PMC5, bits 15-**19** control event selection for PMC6, bits **20**-**24** control event selection for PMC7, and bits 25-**28** control event selection for PMC8.

The counter selection fields, e.g., bits 19-25 and bits **26**-**31** of MMCRO and bits 0-**28** of MMCR1, preferably have as many bits necessary to specify the full domain of selectable events provided by a particular implementation.

Gover, column 11, lines 14-49.

Of course, because of the many paths conditional branching may follow, determinations on a best course of action for improvements are system and situation dependent. For the PowerPC architecture, the type of counts suitable for use in this evaluation includes a count of the number of: instructions dispatched along the branch target path; instructions dispatched sequentially (along the branch target not taken path); basic blocks prefetched; basic blocks incorrectly prefetched; cycles the branch processing unit is idle; cycles the branch unit produces a result or is stalled; branches completed; instructions dispatched to the branch processing unit; accesses to a branch target address cache (BTAC); fetch corrections made at the dispatch stage or at the decode stage; times the condition register (CR) logical unit produced a result; hits or misses in the BTAC; and times all basic block identifiers were used. All of these determinations are utilized to analyze branches to determine the branch unit operation.

Gover, column 20, lines 48-65

In a further examination, the time spent by the processor to execute an instruction indicates system performance efficiency. More particularly, utilizing the performance monitoring facility, counts of the number of instructions entering or exiting per cycle and the number of instructions either entering or leaving the processing system are readily determined. For example, the time required to execute load or store instructions provides valuable system performance data.

Using these counts and the principles of Little's Law, a determination of the average processing time of an instruction is achieved. According to Little's Law, the average time a customer spends in a system equals the average number of customers in the system divided by the average rate that a customer enters or leaves the system. Applying that to the processor system, the average time required to process an instruction, e.g., a load or store, equals the average number of instructions held in the instruction queue divided by the average number of instructions entering the instruction queue. Thus, the counts of the number of instructions and the number of instructions entering the processing system provide the data that is readily used to determine the average time required to process an instruction. In one aspect, the average time is a ratio of the running sum of the number of members for each cycle in the instruction queue to the counted number of members either entering or leaving the instruction queue. Additionally, the method includes altering at least one section of code to reduce the average time the members remain and increase the efficiency of the SP system.

Gover, column 22, lines 8-37

Figures 6A and 6B in Gover illustrate various bit fields and the descriptions thereof reproduced above describe how the bit fields may be set to specify events that are to be counted. As discussed above, these bit fields are not the same as the "data access indicators" recited in claim 1, and, assuming *arguendo*, that they could be so construed, nowhere in any of the above recitations is there any teaching or suggestion of "identifying the data access indicators that have been set by a processor in the data processing system in response to access of the memory locations during execution of the code by the processor to form set data access indicators" as recited in claim 1; or that "each set data access indicator is associated with a portion of the memory locations allocated for the code, and wherein the data access indicators are located in one of a shadow memory or a page table" as also recited in claim 1.

Yet further, Gover also does not disclose "identifying unset data access indicators that have remained unset during execution of the code by the processor" as recited in claim 1. The Examiner refers to column 12, lines 10-11 of Gover which recites "[O]nce the enabled IABR [instruction address breakpoint register] match occurs, bit 29 of MMCR1 [monitor mode control register] is reset and counting occurs." This recitation has nothing to do with "identifying unset data access indicators that have remained unset during execution of the code by the processor" but simply describes resetting of a register and initiation of a counting operation.

For at least all the above reasons, Appellants respectfully submit that Gover fails to disclose or suggest any of the claimed steps of "obtaining the coverage data containing data access indicators associated with memory locations", "identifying the data access indicators that have been set by a processor in the data processing system in response to access of the memory locations during execution of the code by the processor to form set data access indicators, wherein each set data access indicator is associated with a portion of the memory locations allocated for the code, and wherein the data access indicators are located in one of a shadow memory or a page table" or "identifying unset data access indicators that have remained unset during execution of the code by the processor".

Claim 1 further recites the step of "generating a presentation for coverage data in response to an event for identifying memory locations that have been accessed and for identifying memory locations that have not been accessed during execution of the code, wherein the set data access indicators and the unset data access indicators are identified in the presentation by one of using a first color to identify the set data access indicators and using a second color to identify

the unset data access indicators, and using a graphical indicator to identify the set data access indicators and the unset data access indicators." The Examiner acknowledges, and Appellants agree, that Gover also does not disclose or suggest this feature of the claim. The Examiner, however, cites Johnston as teaching this feature. Appellants respectfully disagree.

Johnson is directed to presenting test coverage metrics in a same general visual manner in which a programmer creates a visual program (see the Abstract in Johnston). Although Johnston may disclose displaying information in different colors to facilitate understanding information, Johnston is not related to and does not disclose presenting coverage data relating to monitoring the execution of a computer program, but is instead related to presenting coverage metrics relating to testing software. This is clearly described in Column 1, line 62 to column 2, line 15 of Johnston referred to by the Examiner in rejecting claim 1 and reproduced below for the convenience of the Board:

> Statement coverage is concerned with counting the number of statements of program code that are executed during a test, and comparing this number to the total number of statements in the program. Branch testing, or branch coverage, is concerned with testing each branching statement (e.g. an IF . . . THEN . . . ELSE statement, or a SELECT statement) under conditions that will cause specific branches to be taken. The branch coverage metric then indicates how many of the possible branches in the code have been exercised during the test.
>
> Component-based visual programs introduce a new challenge to measuring code coverage. (The term "visual program" is used herein as a shorthand for "component-based visual program".) Visual programs are different from conventional programs in several important ways. Because of these differences, known test coverage techniques cannot be applied to this new programming domain in a way that provides meaningful results-and without the ability to define test coverage metrics for visual programs, it is difficult to assess program quality and reliability, which may negatively affect the acceptance of such programs by end-users.

Johnston is unrelated to identifying memory locations that have been accessed and memory locations that have not been accessed during execution of code and does not disclose or suggest "generating a presentation for coverage data in response to an event for identifying memory locations that have been accessed and for identifying memory locations that have not been accessed during execution of the code, wherein the set data access indicators and the unset data access indicators are identified in the presentation by one of using a first color to identify the set data access indicators and using a second color to identify the unset data access indicators,

and using a graphical indicator to identify the set data access indicators and the unset data access indicators", and does not supply this acknowledged deficiency in Gover.

For at least all the above reasons, neither Gover nor Johnston nor their combination discloses or suggests the steps recited in claim 1, and the Examiner has not established a *prima facie* case of obviousness in rejecting claim 1.

Independent claims 11 and 21 recite similar subject matter as claim 1, and the Examiner has not established a *prima facie* case of obviousness in rejecting claims 11 and 21 for similar reasons as described in detail above with respect to claim 1.

Claims 3 and 8-10 depend from and further restrict claim 1; claims 13 and 18-20 depend from and further restrict claim 11; and claim 23 depends from and further restricts claim 21. These claims also patentably distinguish over Gover in view of Johnston at least by virtue of their dependency.

A.2. Claims 8 and 18

Claim 8 depends from claim 1 and is as follows:

> 8. The method of claim 1, wherein the event is at least one of a completion of the execution of the code, expiration of a time, and the execution of a selected type of instruction in the code.

The Examiner asserts that Gover discloses the subject matter of claim 8 because Gover "includes a counter that designates a precise point in time for saving the machine state" (see page 7 of Final Office Action dated October 4, 2007). Assuming *arguendo* that the Examiner is correct that Gover teaches such a counter, Gover does not, as acknowledged by the Examiner, disclose generating a presentation for coverage data, and does not disclose or suggest generating a presentation for coverage data in response to at least one of the events recited in claim 8 for identifying memory locations that have been accessed and for identifying memory locations that have not been accessed during execution of the code.

Accordingly, claim 8 and corresponding claim 18 patentably distinguish over Gover in view of Johnston in their own right as well as by virtue of their dependency and the Examiner has not established a *prima facie* case of obviousness in rejecting claims 8 and 18.

A.3. Claims 9 and 19

Claim 9 depends from and further restricts claim 1 and is as follows:

> 9. The method of claim 1, wherein the portion of the memory locations is a single memory location in the code and wherein every memory location in the memory locations is associated with a different data access indicator.

The Examiner asserts that claim 9 is disclosed by Figures 6A and 6B of Gover, reproduced above, and by column 3, lines 24-25 in Gover describing Figures 6A and 6B as illustrating monitor mode control registers (MMCRs) to manage a plurality of counters. Appellants respectfully disagree. Neither Figure 6A nor 6B nor their description teaches or suggests that every memory location in the memory locations of claim 1 is associated with a different data access indicator. As indicated above, Appellants dispute that Gover even discloses a data access indicator, however, the reference certainly does not disclose or suggest a different data access indicator associated with every memory location.

Accordingly, claim 9 and corresponding claim 19 patentably distinguish over Gover in view of Johnston in their own right as well as by virtue of their dependency, and the Examiner has not established a *prima facie* case of obviousness in rejecting claims 9 and 19.

B. CONCLUSION

For at least all the above reasons, the Examiner has not established a *prima facie* case of obviousness in rejecting claims 1, 3, 8-11, 13, 18-21 and 23; and the claims patentably distinguish over Gover in view of Johnston. It is, accordingly, respectfully requested that the Board of Patent Appeals and Interferences reverse the Examiner's Final Rejection of those claims.

/Gerald H. Glanzman/
Gerald H. Glanzman
Reg. No. 25,035
YEE & ASSOCIATES, P.C.
PO Box 802333
Dallas, TX 75380
(972) 385-8777

CLAIMS APPENDIX

The text of the claims involved in the appeal is as follows:

1. A method in a data processing system for presenting coverage data relating to data access occurring during execution of code, the method comprising:

obtaining the coverage data containing data access indicators associated with memory locations;

identifying the data access indicators that have been set by a processor in the data processing system in response to access of the memory locations during execution of the code by the processor to form set data access indicators, wherein each set data access indicator is associated with a portion of the memory locations allocated for the code, and wherein the data access indicators are located in one of a shadow memory or a page table;

identifying unset data access indicators that have remained unset during execution of the code by the processor; and

generating a presentation for coverage data in response to an event for identifying memory locations that have been accessed and for identifying memory locations that have not been accessed during execution of the code, wherein the set data access indicators and the unset data access indicators are identified in the presentation by one of using a first color to identify the set data access indicators and using a second color to identify the unset data access indicators, and using a graphical indicator to identify the set data access indicators and the unset data access indicators.

3. The method of claim 1, wherein the event is completion of the execution of the code, and further comprising:

receiving new test parameters after generating the presentation; and

in response to receiving the new test parameters, repeating the obtaining step, the identifying steps, and the generating step.

8. The method of claim 1, wherein the event is at least one of a completion of the execution of the code, expiration of a time, and the execution of a selected type of instruction in the code.

9. The method of claim 1, wherein the portion of the memory locations is a single memory location in the code and wherein every memory location in the memory locations is associated with a different data access indicator.

10. The method of claim 1, wherein the portion of the memory locations includes at least one of a memory area or a single memory location.

11. A data processing system for presenting coverage data relating to data access occurring during execution of code, the data processing system comprising:

obtaining means for obtaining the coverage data containing data access indicators associated with memory locations;

first identifying means for identifying the data access indicators that have been set by a processor in the data processing system in response to access of the memory locations during execution of the code by the processor to form set data access indicators, wherein each set data

access indicator is associated with a portion of the memory locations allocated for the code, and wherein the data access indicators are located in one of a shadow memory or a page table;

second identifying means for identifying unset data access indicators that have remained unset during execution of the code by the processor; and

generating means for generating a presentation for coverage data in response to an event for identifying memory locations that have been accessed and for identifying memory locations that have not been accessed during execution of the code, wherein the set data access indicators and the unset data access indicators are identified in the presentation by one of using a first color to identify the set data access indicators and using a second color to identify the unset data access indicators, and using a graphical indicator to identify the set data access indicators and the unset data access indicators.

13. The data processing system of claim 11, wherein the event is completion of the execution of the code, and further comprising:

receiving means for receiving new test parameters after generating the presentation; and

repeating means, responsive to receiving the new test parameters, for repeating the obtaining means, the first and second identifying means, and the generating means.

18. The data processing system of claim 11, wherein the event is at least one of a completion of the execution of the code, expiration of a time, and the execution of a selected type of instruction in the code.

19. The data processing system of claim 11, wherein the portion of the memory locations is a single memory location in the code and wherein every memory location in the memory locations is associated with a different data access indicator.

20. The data processing system of claim 11, wherein the portion of the memory locations includes at least one of a memory area or a single memory location.

21. A computer program product in a recordable-type computer readable medium for presenting coverage data relating to data access occurring during execution of code, the computer program product comprising:

first instructions for obtaining the coverage data containing data access indicators associated with memory locations;

second instructions for identifying the data access indicators that have been set by a processor in the data processing system in response to access of the memory locations during execution of the code by the processor to form set data access indicators, wherein each set data access indicator is associated with a portion of the memory locations allocated for the code, and wherein the data access indicators are located in one of a shadow memory or a page table;

third instructions for identifying unset data access indicators that have remained unset during execution of the code by the processor; and

fourth instructions for generating a presentation for coverage data in response to an event for identifying memory locations that have been accessed and for identifying memory locations that have not been accessed during execution of the code, wherein the set data access indicators and the unset data access indicators are identified in the presentation by one of using a first color

to identify the set data access indicators and using a second color to identify the unset data access indicators, and using a graphical indicator to identify the set data access indicators and the unset data access indicators.

23. The computer program product of claim 21, wherein the event is completion of the execution of the code, and further comprising:

fifth instructions for receiving new test parameters after generating the presentation; and

sixth instructions, responsive to receiving the new test parameters, for repeating the first instructions, the second instructions, the third instructions and the fourth instructions.

EVIDENCE APPENDIX

This appeal brief presents no additional evidence.

RELATED PROCEEDINGS APPENDIX

This appeal has no related proceedings.